快速傅里叶变换

及生成函数相关运用

罗成元
巴黎卓越工程师学院
2022 年 12 月 3 日

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录

复数

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

定义

形如 $a+ib\ (a,b\in\mathbb{R}^2)$ 的数称为复数，其中 i 为虚数单位，$i^2=-1$。

- 复数的加法运算：

$$(a+ib)+(c+id)=(a+c)+i(b+d)$$

- 复数的加法满足封闭性、交换律、结合律，存在加法零元 0 $(\forall x\in\mathbb{C},\ x+0=x)$，并且存在加法逆元 $(\forall x\in\mathbb{C},\ \exists y\in\mathbb{C},\ x+y=y+x=0)$。

复数

定义

形如 $a+ib\ (a,b\in\mathbb{R}^2)$ 的数称为复数，其中 i 为虚数单位，$i^2=-1$。

- 复数的乘法运算：

$$(a+ib)(c+id)=ac-bd+i(ad+bc)$$

- 复数的乘法满足封闭性、交换律、结合律、分配律，存在乘法幺元 1 $(\forall x\in\mathbb{C},\ 1x=x1=x)$，存在乘法逆元 $(\forall x\in\mathbb{C}^*,\ \exists y\in\mathbb{C},\ xy=yx=1)$。

复数

- 设 $z = a + ib$。

复数的模长

$$|z| = \sqrt{a^2 + b^2}$$

- 如果 a, b 不全为 0，
$$z = \frac{a}{\sqrt{a^2+b^2}} + i\frac{b}{\sqrt{a^2+b^2}}$$
- 可以发现，存在 θ 使得 $\cos(\theta) = \frac{a}{\sqrt{a^2+b^2}}$，$\sin(\theta) = \frac{b}{\sqrt{a^2+b^2}}$。
- 因此可以把 $z = a + ib$ 写成 $|z|(\cos(\theta) + i\sin(\theta))$。这是复数的极坐标表示形式。

欧拉公式

$$e^{i\theta} = \cos(\theta) + i\sin(\theta)$$

- 所以可以把 z 写成 $|z|e^{i\theta}$ 的形式。

单位根

n 次单位根

方程

$$z^n = 1 \ (n \in \mathbb{N}^*)$$

的复数根 z 为 n 次单位根。

- 可以验证 n 次单位根共有 n 个：
$$e^{i\frac{2\pi}{n}k} \ (k = 0, 1, \ldots, n-1)$$
- 因为
$$\left(e^{i\frac{2\pi}{n}k}\right)^n = e^{i2\pi k} = \cos(2\pi k) + i\sin(2\pi k) = 1$$
- 设 $\omega_n = e^{i\frac{2\pi}{n}}$。性质：
  1. $\omega_n^k = \omega_{2n}^{2k}$
  2. $\omega_n^k = \omega_n^{k \bmod n}$

傅里叶变换

- 大二上 Signaux（信号学）第四章中：

傅里叶变换

对于一个周期为 T 的函数 f，求傅里叶级数：

$$\tilde{C}_n = \frac{2}{T}\int_{t_0}^{t_0+T} f(t)\exp\left(-j\frac{2\pi}{T}nt\right)\mathrm{d}t.$$

可以推得

$$f(t) = \mathrm{Re}\left(\sum_{n=0}^{+\infty}\tilde{C}_n\exp\left(j\frac{2\pi}{T}nt\right)\right)$$

离散傅里叶变换

- **离散傅里叶变换**：Discrete Fourier Transform（Transformation de Fourier discrète）

离散傅里叶变换

对于一个序列 $(s_n)_{n\in[\![0,N-1]\!]}$，其离散傅里叶变换后的序列 (S_n) 满足：
$\forall\ n \in [\![0, N-1]\!]$,

$$S_n = \sum_{k=0}^{N-1} s_k \exp\left(-i\frac{2\pi}{N}kn\right)$$

离散傅里叶变换

连续傅里叶变换	离散傅里叶变换
$\tilde{C}_n = \frac{2}{T}\int_{t_0}^{t_0+T} f(t)\exp\left(-jn\omega t\right)\mathrm{d}t$	$S_n = \sum_{k=0}^{N-1} s_k \exp\left(-i\frac{2\pi}{N}kn\right)$
T	N
n	n
t	k

离散傅里叶变换

离散傅里叶逆变换

逆变换（Inverse DFT，或 IDFT）

$$s_k = \frac{1}{N} \sum_{n=0}^{N-1} S_n \exp\left(i \frac{2\pi}{N} kn\right)$$

离散傅里叶变换

连续傅里叶变换	离散傅里叶变换
$\tilde{C_n} = \frac{2}{T}\int_{t_0}^{t_0+T} f(t)\exp\left(-jn\omega t\right)\mathrm{d}t$	$S_n = \sum_{k=0}^{N-1} s_k \exp\left(-i\frac{2\pi}{N}kn\right)$
T	N
n	n
t	k
$f(t) = \mathrm{Re}\left(\sum_{n=0}^{+\infty}\tilde{C_n}\exp\left(j\frac{2\pi}{T}nt\right)\right)$	$s_k = \frac{1}{N}\sum_{n=0}^{N-1} S_n \exp\left(i\frac{2\pi}{N}kn\right)$
T	N
t	k
n	n

离散傅里叶变换

- 证明逆变换：
- 设 $\omega = \exp\left(i\frac{2\pi}{N}\right)$。$\forall\ k \in \mathbb{Z}$，

$$\sum_{n=0}^{N-1} \omega^{kn} = \sum_{n=0}^{N-1} (\omega_k)^n = \begin{cases} N, & N \mid k, \\ \frac{1-\omega^{kN}}{1-\omega^k} = \frac{1-\exp\left(i\frac{2\pi}{N}kN\right)}{1-\omega^k} = 0, & N \nmid k \end{cases}$$

- $\forall\ 0 \le k < N$，

$$\begin{aligned} s_k &= \frac{1}{N}\sum_{n=0}^{N-1} \omega^{kn} S_n = \frac{1}{N}\sum_{n=0}^{N-1} \omega^{kn} \sum_{r=0}^{N-1} \omega^{-rn} s_r \\ &= \frac{1}{N}\sum_{r=0}^{N-1} s_r \sum_{n=0}^{N-1} \omega^{(k-r)n} = \frac{1}{N}\sum_{r=0}^{N-1} s_r [k=r] N = s_k \end{aligned}$$

离散傅里叶变换

- $\forall\ 0 \le n < N$,

$$
\begin{aligned}
S_n &= \sum_{k=0}^{N-1} \omega^{-kn} s_k = \sum_{k=0}^{N-1} \omega^{-kn} \frac{1}{N} \sum_{r=0}^{N-1} \omega^{kr} S_r \\
&= \sum_{r=0}^{N-1} S_r \frac{1}{N} \sum_{k=0}^{N-1} \omega^{(r-n)k} = \sum_{r=0}^{N-1} S_r \frac{1}{N} [r=n] N = S_n
\end{aligned}
$$

$$
\mathrm{IDFT} \circ \mathrm{DFT} = \mathrm{DFT} \circ \mathrm{IDFT} = \mathrm{id}_{\mathbb{C}^{[\![0,N-1]\!]}}.
$$

快速傅里叶变换

- 如果要根据定义进行 DFT，对于每一个 $n \in [\![0, N-1]\!]$，计算 $S_n = \sum_{k=0}^{N-1} \omega^{-kn} s_k$，那么需要先枚举 $n \in [\![0, N-1]\!]$，再枚举 $k \in [\![0, N-1]\!]$，时间复杂度 $O(N^2)$。IDFT 同理。
- 存在更快的方法，快速傅里叶变换，Fast Fourier Transform（FFT）。
- 其中比较常见的是 Cooley-Tukey 算法。用分治的思想，时间复杂度提升到 $O(N \log N)$（$\log$ 以 2 为底）。

Cooley-Tukey

- 观察 $N=4$ 的情况 s_0, s_1, s_2, s_3 $\left(\omega = \exp\left(i\frac{2\pi}{4}\right)\right)$。
- 我们将 (s_0, s_1, s_2, s_3) 分成两个序列：$(t_0 = s_0, t_1 = s_2)$, $(t'_0 = s_1, t'_1 = s_3)$，并分别对两个序列进行傅里叶变换，得到 $T_0 = s_0 + s_2$, $T_1 = s_0 + \omega^{-2} s_2$, $T'_0 = s_1 + s_3$, $T'_1 = s_1 + \omega^{-2} s_3$.
- 现在要计算出 (s_0, s_1, s_2, s_3) 的快速傅里叶变换：

$$\begin{cases} S_0 = s_0 + s_1 + s_2 + s_3 = T_0 + T'_0, \\ S_2 = s_0 + \omega^{-2} s_1 + \omega^{-4} s_2 + \omega^{-6} s_3 = T_0 + \omega^{-2} T'_0 = T_0 - T'_0 \\ S_1 = s_0 + \omega^{-1} s_1 + \omega^{-2} s_2 + \omega^{-3} s_3 = T_1 + \omega^{-1} T'_1, \\ S_3 = s_0 + \omega^{-3} s_1 + \omega^{-6} s_3 + \omega^{-9} s_3 = T_1 + \omega^{-3} T'_1 = T_1 - \omega^{-1} T'_1. \end{cases}$$

Cooley-Tukey

- 再扩展到 $N = 8$ 的情况（$\omega = \exp\left(i\frac{2\pi}{8}\right)$）。

$$
\begin{cases}
S_0 = (s_0 + s_1 + s_2 + s_3) + (s_4 + s_5 + s_6 + s_7) = T_0 + T_0', \\
S_4 = (s_0 + s_2 + s_4 + s_6) + \omega^{-4}(s_1 + s_3 + s_5 + s_7) = T_0 + \omega^{-4}T_0' = T_0 - T_0', \\
S_1 = (s_0 + \omega^{-2}s_2 + \omega^{-4}s_4 + \omega^{-6}s_6) + \omega^{-1}(s_1 + \omega^{-2}s_3 + \omega^{-4}s_5 + \omega^{-6}s_7) = T_1 + \omega^{-1}T_1', \\
S_5 = (s_0 + \omega^{-2}s_2 + \omega^{-4}s_4 + \omega^{-6}s_6) + \omega^{-5}(s_1 + \omega^{-2}s_3 + \omega^{-4}s_5 + \omega^{-6}s_7) = T_1 - \omega^{-1}T_1', \\
S_2 = (s_0 + \omega^{-4}s_2 + s_4 + \omega^{-4}s_6) + \omega^{-2}(s_1 + \omega^{-4}s_3 + s_5 + \omega^{-4}s_7) = T_2 + \omega^{-2}T_2', \\
S_6 = (s_0 + \omega^{-4}s_2 + s_4 + \omega^{-4}s_6) + \omega^{-6}(s_1 + \omega^{-4}s_3 + s_5 + \omega^{-4}s_7) = T_2 - \omega^{-2}T_2', \\
S_3 = (s_0 + \omega^{-6}s_2 + \omega^{-4}s_4 + \omega^{-2}s_6) + \omega^{-3}(s_1 + \omega^{-6}s_3 + \omega^{-4}s_5 + \omega^{-2}s_7) = T_3 + \omega^{-3}T_3', \\
S_7 = (s_0 + \omega^{-6}s_2 + \omega^{-4}s_4 + \omega^{-2}s_6) + \omega^{-7}(s_1 + \omega^{-6}s_3 + \omega^{-4}s_5 + \omega^{-2}s_7) = T_3 - \omega^{-3}T_3'.
\end{cases}
$$

Cooley-Tukey

- 假设 $N = 2^r$。设 $M = \frac{N}{2} = 2^{r-1}$。
- 已经分别对 $(s_0, s_2, \ldots, s_{N-2})$ 和 $(s_1, s_3, \ldots, s_{N-1})$ 进行傅里叶变换，得到 $(T_0, T_1, \ldots, T_{M-1})$ 和 $(T'_0, T'_1, \ldots, T'_{M-1})$。

$$T_k = \sum_{n=0}^{M-1} s_{2n} \exp\left(-i\frac{2\pi}{M}kn\right)$$

$$T'_k = \sum_{n=0}^{M-1} s_{2n+1} \exp\left(-i\frac{2\pi}{M}kn\right)$$

Cooley-Tukey

$$
\begin{aligned}
S_k &= \sum_{n=0}^{N-1} s_n \exp\left(-i\frac{2\pi}{N}kn\right) \\
&= \sum_{n=0}^{M-1} s_{2n} \exp\left(-i\frac{2\pi}{N}k\cdot 2n\right) + \sum_{n=0}^{M-1} s_{2n+1} \exp\left(-i\frac{2\pi}{N}k\cdot(2n+1)\right) \\
&= \sum_{n=0}^{M-1} s_{2n} \exp\left(-i\frac{2\pi}{M}kn\right) + \exp\left(-i\frac{2\pi}{N}k\right) \sum_{n=0}^{M-1} s_{2n+1} \exp\left(-i\frac{2\pi}{M}kn\right) \\
&= T_k + T'_k \exp\left(-i2\pi\frac{k}{N}\right)
\end{aligned}
$$

- 或者:

$$
S_k = \begin{cases} T_k + T'_k \exp\left(-i2\pi\frac{k}{N}\right), & k < M, \\ T_{k-M} - T'_{k-M} \exp\left(-i2\pi\frac{k-M}{N}\right), & k \geq M. \end{cases}
$$

蝴蝶变换

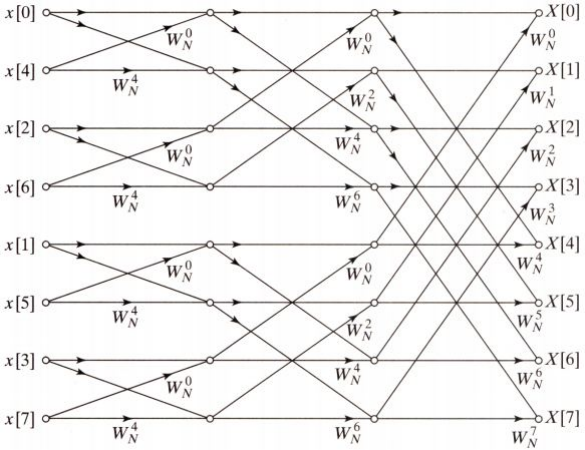

图: 蝴蝶变换[1]

伪代码

func dft($s_0, s_1, \ldots, s_{N-1}$):
  Result: $(S_0, S_1, \ldots, S_{N-1})$
  if $N = 1$ **then**
    return (s_0)
  else
    $M \leftarrow \frac{N}{2}$
    $(T_0, T_1, \ldots, T_{M-1}) \leftarrow$ dft$(s_0, s_2, \ldots, s_{N-2})$
    $(T'_0, T'_1, \ldots, T'_{M-1}) \leftarrow$ dft$(s_1, s_3, \ldots, s_{N-1})$
    for $k \leftarrow 0$ **to** $M - 1$ **do**
      $S_k \leftarrow T_k + T_{k+M} \exp\left(-i2\pi \frac{k}{N}\right)$
      $S_{k+M} \leftarrow T_k - T_{k+M} \exp\left(-i2\pi \frac{k}{N}\right)$
    end
    return $(S_0, S_1, \ldots, S_{N-1})$
  end

Cooley-Tukey

- 对于离散傅里叶逆变换：只需要把 $\exp\left(-i2\pi\frac{k}{N}\right)$ 改为 $\exp\left(i2\pi\frac{k}{N}\right)$。

func `idft`$(s_0, s_1, \ldots, s_{N-1})$**:**
  Result: $(S_0, S_1, \ldots, S_{N-1})$
  if $N = 1$ **then**
    return (s_0)
  else
    $M \leftarrow \frac{N}{2}$
    $(T_0, T_1, \ldots, T_{M-1}) \leftarrow$ `idft`$(s_0, s_2, \ldots, s_{N-2})$
    $(T'_0, T'_1, \ldots, T'_{M-1}) \leftarrow$ `idft`$(s_1, s_3, \ldots, s_{N-1})$
    for $k \leftarrow 0$ **to** $M - 1$ **do**
      $S_k \leftarrow T_k + T_{k+M} \exp\left(+i2\pi\frac{k}{N}\right)$
      $S_{k+M} \leftarrow T_k - T_{k+M} \exp\left(+i2\pi\frac{k}{N}\right)$
    end
    return $(S_0, S_1, \ldots, S_{N-1})$
  end

Cooley-Tukey

- 分析其时间复杂度。
- 设 $T(n)$ 表示对长度为 n 的数列进行 FFT 的计算次数。
- $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$ 其中 $O(n)$ 表示计算次数可以近似为一个常数乘上 n。
- 由主定理得[2] $T(n) = O(n \log n)$，相比于 $O(n^2)$ 时间效率大幅提高。
- 这是以 2 基底的 FFT（每次分成两半），也可以以其他数为基底，或者混合基底。

多项式乘法

- 再观察傅里叶变换公式

$$\begin{aligned} S_n &= \sum_{k=0}^{N-1} s_k \exp\left(-i\frac{2\pi}{N}kn\right) \\ &= \sum_{k=0}^{N-1} s_k \left(\exp\left(-i\frac{2\pi}{N}n\right)\right)^k \end{aligned}$$

- 设

$$F(x) = \sum_{i=0}^{N-1} s_i x^i$$

- 那么

$$S_n = F\left(\exp\left(-i\frac{2\pi}{N}n\right)\right)$$

多项式乘法

- 所以离散傅里叶变换能得到一个函数在 $x = \exp\left(-i\frac{2\pi}{N}n\right)\ (0 \le n < N)$ 处的取值。
- 离散傅里叶逆变换能根据 $x = \exp\left(-i\frac{2\pi}{N}n\right)$ 的取值变换出原来的函数。
- 一个 N 阶的多项式函数，只需要知道在 $N+1$ 个点的取值，就可以确定这个多项式。
- 对于一个 M 阶函数 $F(x)$ 和 R 阶函数 $G(x)$，那么 $(FG)(x) = F(x)G(x)$ 是一个 $M+R$ 阶的函数。
- 我们取 $N = 2^{\lceil \log_2 (M+R+1) \rceil}$。进行离散傅里叶变换得出 F 和 G 在 $x = \exp\left(-i\frac{2\pi}{N}n\right)\ (0 \le n < N)$ 的取值。
- $\forall x \in \mathbb{C}$，$(FG)(x) = F(x)G(x)$。也就得出了 FG 在 $x = \exp\left(-i\frac{2\pi}{N}n\right)$ 处的取值。逆变换后得到 FG 的多项式。

多项式乘法

- 等价于快速算出两个数列的卷积：

$$c_n = \sum_{k=0}^{n} a_k b_{n-k}$$

- 时间复杂度为 $O(n \log n)$。

多项式乘法

- 在实际数值计算中，因为单位根的实部和虚部都为小数，误差不可避免。
- 在信息学竞赛中，为了消除误差的影响，一般取 $p = 998244353$ 为模数：
  - 998244353 是一个质数，$\mathbb{Z}/p\mathbb{Z}$ 是一个域（可以定义加减乘除，任意一个非零的 x 有乘法逆元 x^{-1} 使得 $x \cdot x^{-1} \equiv 1 \pmod p$，因为根据裴蜀定理，$x \cdot x^{-1} - p \cdot y = 1$ 有解）。
  - 998244353 有原根 3（$\forall r \in [\![1, p-2]\!]$，$3^r \not\equiv 1 \pmod p$）。因此 3 满足复数单位根的性质。（欧拉定理：$\forall (a, p) = 1$，$a^{\varphi(p)} \equiv 1 \pmod p$）
  - $998244353 = 7 \times 17 \times 2^{23} + 1$，$2^{23} \mid \varphi(p) = p - 1$。对于任意 $N = 2^r (r \leq 23)$，可以计算出 $3^{\frac{p-1}{N}}$ 作为复数单位根。
- 快速数论变换（Number-theoretic transform）。
- 也有其他模数的快速数论变换算法，如以毛啸命名的 MTT[3]。

多项式求逆

- 对于 $N \in \mathbb{N}^*$ 和一个多项式 $P(x)$ $(p_0 \neq 0)$，求出多项式 $Q(x)$ 使得 $P(x)Q(x) \equiv 1 \pmod{x^N}$。
- 设 $p_0, p_1, \ldots, p_{N-1}$ 为 $P(x)$ 中 $x^0, x^1, \ldots, x^{N-1}$ 次项系数。$q_0, q_1, \ldots, q_{N-1}$ 同理。

第一种方法

- $q_0 = \frac{1}{p_0}$;
- 递推求 q_k $(1 \leq k < N)$:

$$\sum_{i=0}^{k} p_i q_{k-i} = p_0 q_k + \sum_{i=1}^{k} p_i q_{k-i} = 0$$

$$q_k = -\frac{\sum_{i=1}^{k} p_i q_{k-i}}{p_0}$$

- 算法复杂度为 $O(N^2)$。

多项式求逆：第二种方法

- 上式是一个卷积的形式。
- 我们希望用 FFT 优化，但是需要先求出 $q_0, q_1, \ldots, q_{i-1}$ 的值才能求出 q_i。
- 我们采用 CDQ 分治（陈丹琦分治[4]）的思想，批量求解并进行卷积。

用分治进行多项式求逆

- 例如我们现在需要计算 $q_0, q_1, \ldots, q_n$。
- 我们记 $t_k = \sum_{i=1}^{k} p_i q_{k-i}$。那么 $q_k = -\frac{t_k}{p_0}$。所以关键是计算 t_k。
- 假设我们已经计算出 $q_0, q_1, \ldots, q_m$（这一步可以认为是递归求解）。
- 我们先考虑 $q_0, \ldots, q_m$ 对 t_k $(m+1 \le k \le n)$ 的贡献。即计算 $\sum_{1 \le i \le k, k-i \le m} p_i q_{k-i}$。这显然是个卷积的形式，可以 FFT 加速。
- 我们再递归求解 $q_{m+1}, q_{m+2}, \ldots, q_n$。

多项式求逆：第二种方法

CDQ 分治

设一个数列 $(f_0, f_1, \ldots, f_n)$ 存在递推关系。

func `solve`(l, r)**:**

- **Result:** 求出 $f_l, f_{l+1}, \ldots, f_r$
- **if** $l < r$ **then**
  - $mid \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor$
  - `solve`(l, mid)
  - 计算出 $f_l, \ldots, f_{mid}$ 对 $f_{mid+1}, \ldots, f_r$ 的影响
  - `solve`$(mid + 1, r)$
- **else**
  - 求出 f_l
- **end**

多项式求逆：第二种方法

- 正确性：$\forall i$，在求出 f_i 之前，考虑到了 $f_0, f_1, \ldots, f_{i-1}$ 对 f_i 的影响。
- 时间效率：
  - $f_l, \ldots, f_{mid}$ 对 $f_{mid+1}, \ldots, f_r$ 的影响是一个卷积，因此可以用 FFT 加速计算。
  - 时间复杂度 $T(n) = 2T\left(\frac{n}{2}\right) + O(n \log n)$。根据主定理[2]得 $T(n) = O(n \log^2 n)$。

多项式求逆：第三种方法

泰勒展开

- 若一个函数 $f(x)$ 在其定义域一点 x_0 处 n 阶可导，那么

$$\begin{aligned} f(x) &= f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \dots \\ &\quad + \frac{f^{(n)(x_0)}}{n!}(x - x_0)^n + o_{x \to x_0}\left((x - x_0)^n\right) \\ &= \sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!}(x - x_0)^i + o_{x \to x_0}\left((x - x_0)^n\right) \end{aligned}$$

- 其意义是，在 x_0 的领域，可以将 $f(x)$ 拟合成一个多项式。
- 可以验证：原函数和泰勒展开的函数的 0 到 n 阶导是相等的。

多项式求逆：第三种方法

牛顿迭代

- 我们要找出一个函数 $f(x)$ 的零点 $f(x_0) = 0$。
- 用迭代法不断逼近零点。
- 设当前迭代得到的点是 x_1。对 f 在 x_1 处进行一阶泰勒展开。

$$f(x) = f(x_1) + f'(x_1)(x - x_1) + o_{x \to x_1}(x - x_0)$$

求解

$$f(x_1) + f'(x_1)(x - x_1) = 0$$

$$x = x_1 - \frac{f(x_1)}{f'(x_1)}$$

- $x_1 \leftarrow x_1 - \frac{f(x_1)}{f'(x_1)}$ 再进行迭代。

多项式求逆：第三种方法

- 现在要求 $Q(x) \equiv (P(x))^{-1} \pmod{x^N}$（之后省略 (x)）。
- 我们设函数 $F(A) = \frac{1}{A} - P$。那么 $F(Q) \equiv 0 \pmod{x^N}$。
- 考虑牛顿迭代：假设我们求出了 $Q_0 \equiv P^{-1} \pmod{x^{\lceil N/2 \rceil}}$。现在要求 $Q_1 \equiv P^{-1} \pmod{x^N}$，即求出 P^{-1} 的第 $\lceil N/2 \rceil$ 至第 $N-1$ 项系数。
- $Q_0 \equiv Q_1 \pmod{x^{\lceil N/2 \rceil}}$。

$$
\begin{aligned}
F(Q_1) &\equiv F(Q_0) + F'(Q_0)(Q_1 - Q_0) + \frac{F''(Q_0)}{2}(Q_1 - Q_0)^2 + \dots \\
&\equiv F(Q_0) + F'(Q_0)(Q_1 - Q_0) \equiv 0 \pmod{x^N} \\
Q_1 &\equiv Q_0 - \frac{F(Q_0)}{F'(Q_0)} \pmod{x^N}
\end{aligned}
$$

多项式求逆：第三种方法

- $F'(A) = \frac{\mathrm{d}F}{\mathrm{d}A}(A) = -\frac{1}{A^2}$

$$Q_1 \equiv Q_0 + Q_0^2\left(\frac{1}{Q_0} - P\right) \pmod{x^N}$$

$$Q_1 \equiv 2Q_0 - PQ_0^2 \pmod{x^N}$$

- **可以验证**：$PQ_1 \equiv P(2Q_0 - PQ_0)^2 \equiv PQ_0(2 - PQ_0) \pmod{x^N}$
- $PQ_0 \equiv 1 \pmod{x^{\lceil N/2 \rceil}}$。**所以存在一个多项式** R **使得** $PQ_0 \equiv 1 + x^{\lceil N/2 \rceil}R \equiv 1 \pmod{x^N}$。
- $PQ_1 \equiv \left(1 + x^{\lceil N/2 \rceil}R\right)\left(1 - x^{\lceil N/2 \rceil}R\right) \equiv 1 - \left(x^{\lceil N/2 \rceil}R\right)^2 \equiv 1 \pmod{x^N}$

多项式求逆：第三种方法

- 伪代码：

func `getInv`(P, N)**:**

  Result: $Q \equiv P^{-1} \pmod{x^N}$

  if $N = 1$ **then**

    return p_0^{-1}

  else

    $M \leftarrow \lceil \frac{N}{2} \rceil$

    $Q_0 \leftarrow$ `getInv`$(P \bmod x^M, M)$

    return $Q_0(2 - PQ_0) \bmod x^N$

  end

- 时间复杂度 $T(N) = T\left(\frac{N}{2}\right) + O(N \log N)$。根据主定理[2]得 $T(N) = O(N \log N)$。

多项式除法

- 给定 N 阶多项式 A 和 M $(M < N)$ 阶多项式 B $(a_N, b_M \neq 0)$。求多项式 Q 和 R 使得 $A = BQ + R$ 且 $\deg(R) < M$。
- 可以得出 Q 是一个 $N - M$ 阶多项式，且 $q_{N-M} = \frac{a_N}{b_M}$，随后可以递推出 $q_{N-M-1}, q_{N-M-2}, \ldots, q_0$。求出 Q 后即可算出 $R = A - BQ$。
- 观察到这里是从高到低确定系数。如果沿用多项式求逆的思路，就需要将这两个多项式的系数颠倒。
- $\bar{A}(x) = x^N A\left(\frac{1}{x}\right) = x^N \sum_{i=0}^{N} a_i x^{-i} = \sum_{i=0}^{N} a_i x^{N-i} = \sum_{j=0}^{N} a_{N-j} x^j$。
- 同理，$\bar{B}(x) = x^M B\left(\frac{1}{x}\right)$，$\bar{Q}(x) = x^{N-M} Q\left(\frac{1}{x}\right)$ 且 $\bar{R}(x) = x^{M-1} Q\left(\frac{1}{x}\right)$。

多项式除法

$$
\begin{aligned}
A(x) &= B(x)Q(x) + R(x) \\
x^N A\left(\frac{1}{x}\right) &= x^M B\left(\frac{1}{x}\right) x^{N-M} Q\left(\frac{1}{x}\right) + x^{N-M+1} \cdot x^{M-1} R\left(\frac{1}{x}\right) \\
\bar{A}(x) &= \bar{B}(x)\bar{Q}(x) + x^{N-M+1}\bar{R}(x) \\
\bar{A}(x) &\equiv \bar{B}(x)\bar{Q}(x) \pmod{x^{N-M+1}} \\
\bar{Q}(x) &\equiv \bar{A}(x) \cdot (\bar{B}(x))^{-1} \pmod{x^{N-M+1}}
\end{aligned}
$$

- 可以使用多项式求逆求得 $\bar{Q}(x)$，再颠倒系数得到 $Q(x)$。时间复杂度为 $O(N \log N)$。

多项式快速幂

- **多项式 ln**：**对于一个多项式** $P(x)$ $(p_0 = 1)$，**求** $\ln P(x)$。
  - $(\ln P(x))' = \frac{P'(x)}{P(x)}$
  - $\ln P(x) = \int \frac{P'(x)}{P(x)} \mathrm{d}x$ $([x^0] \ln P(x) = 0)$
  - $\ln P(x) \equiv \int P'(x)(P(x))^{-1} \mathrm{d}x \pmod{x^N}$
- **多项式 exp**：**对于一个多项式** $P(x)$ $(p_0 = 0)$，**求** $\exp(P(x)) \pmod{x^N}$：
  - $Q(x) \equiv \exp(P(x)) \equiv \sum_{i=0}^{+\infty} \frac{P(x)^i}{i!} \equiv \sum_{i=0}^{N-1} \frac{P(x)^i}{i!} \pmod{x^N}$
  - **设** $F(A) = \ln(A) - P$。**所以** $F'(A) = \frac{1}{A}$。
  - **牛顿迭代法**：$Q_1 \equiv Q_0 - Q_0 (\ln(Q_0) - P) \pmod{x^N}$。
- **多项式快速幂**：**求** $P(x)^k \pmod{x^N}$。
  - **设** $P'(x) = \frac{P(x)}{p_0}$。**则** $[x^0]P'(x) = 1$，$P(x) = p_0 P'(x)$。
  - $P(x)^k = p_0^k P'(x)^k = p_0^k \exp(k \ln P'(x))$。

多项式多点求值

- 对于一个 N 阶多项式函数 $P(x)$。
- 给定 M 个点 $(x_1, x_2, \ldots, x_M)$，求出 $P(x_1), P(x_2), \ldots, P(x_M)$。
- 事实上，$P(x_k) = P(x) \bmod (x - x_k)$。
- 证明：
  - 设 $Q(x), R$ 满足 $Q(x)(x - x_k) + R = P(x)$。
  - 代入 x_k，$R = Q(x_k)(x_k - x_k) + R = P(x_k)$。

多项式多点求值

- 可以分治求解。
- 先通过分治计算 $\prod_{i=1}^{M}(x-x_i)$，并计算出 $P(x) \bmod \left(\prod_{i=1}^{M}(x-x_i)\right)$。
- 再自上而下分治。如 $\texttt{solve}(l,r)$，已有 $P(x) \bmod \left(\prod_{i=l}^{r}(x-x_i)\right)$（记为 $P_{l,r}(x)$）。
- 设 $mid = \left\lfloor \frac{l+r}{2} \right\rfloor$。
- 计算 $P_{l,mid} = P_{l,r}(x) \bmod \left(\prod_{i=l}^{mid}(x-x_i)\right)$，并继续对 (l, mid) 分治求解。同理对 $(mid+1, r)$ 分治求解。
- 到最后 $l = r$ 时，就得到了该多项式函数在 $x = x_l$ 处的取值。

多项式多点求值

func `getProd`$(l,\ r)$**:**
  if $l < r$ **then**
    $mid \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor$
    return $Prod_{l,r} \leftarrow$ `getProd`$(l,\ mid) \times$ `getProd`$(mid+1,\ r)$
  else
    return $Prod_{l,l} \leftarrow x - x_l$
  end
func `getValue`$(l,\ r,\ P)$**:**
  if $l < r$ **then**
    $mid \leftarrow \left\lfloor \frac{l+r}{2} \right\rfloor$
    `getValue`$(l,\ mid,\ P \bmod Prod_{l,mid})$
    `getValue`$(mid+1,\ r,\ P \bmod Prod_{mid+1,r})$
  else
    $res_l \leftarrow P$
  end

多项式多点求值

- 分析复杂度：
- `getProd` 和 `getValue`：$T(M) = 2T\left(\frac{M}{2}\right) + O(M \log M) = O(M \log^2 M)$。
- 总时间复杂度为 $O(N \log N + M \log^2 M)$。

多项式快速插值

- 给定 $N+1$ 个点 $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$（横坐标互不相等）。确定一个 N 阶多项式函数 P 使得 $\forall 0 \le i \le n,\ P(x_i) = y_i$。

拉格朗日插值

- 设

$$Q_k(x) = \prod_{0 \le i \le N, i \ne k} (x - x_i)$$

- 唯一的 N 阶多项式函数 P 满足

$$P(x) = \sum_{k=0}^{N} \frac{y_k}{Q_k(x_k)} Q_k(x)$$

多项式快速插值

- 首先我们解决如何计算 $Q_k(x_k)$。
- $Q_k(x) = \frac{\prod_{i=0}^{N}(x-x_i)}{x-x_k}$。
- 当 $x = x_k$ 时，$Q_k(x)$ 分子分母取值都为 0。
- 洛必达法则：$Q_k(x_k) = \frac{\left(\prod_{i=0}^{N}(x-x_i)\right)'(x_k)}{(x-x_k)'(x_k)} = \left(\prod_{i=0}^{N}(x-x_i)\right)'(x_k)$。
- $\left(\prod_{i=0}^{N}(x-x_i)\right)'$ 不取决于 k，因此可以 $O(N\log^2 N)$ 多点求值。

多项式快速插值

- 现在问题在于如何快速求出 $\sum_{k=0}^{N} y'_k Q_k(x)$。
- 可以把 $Q_k(x)$ 拆为 $Prod_{0,k-1}(x) = \prod_{i=0}^{k-1}(x - x_i)$ 和 $Prod_{k+1,N}(x) = \prod_{i=k+1}^{N}(x - x_i)$。
- 我们可以在分治到 $[l, r]$ 时，同时求出 $\prod_{k=l}^{r}(x - x_k)$ 和 $\sum_{k=l}^{r} y'_k \prod_{l \le i \le r, i \ne k}(x - x_i)$。

多项式快速插值

- 合并 $[l, mid]$ 和 $[mid+1, r]$ 的答案：

$$\prod_{k=l}^{r}(x-x_k)=\prod_{k=l}^{mid}(x-x_k)\cdot\prod_{k=mid+1}^{r}(x-x_k)$$

$$\sum_{k=l}^{r}y'_k\prod_{l\le i\le r,i\ne k}(x-x_i)=\left(\sum_{k=l}^{mid}y'_k\prod_{l\le i\le mid,i\ne k}(x-x_i)\right)\prod_{i=mid+1}^{r}(x-x_i)$$
$$+\left(\sum_{k=mid+1}^{r}y'_k\prod_{mid+1\le i\le r,i\ne k}(x-x_i)\right)\prod_{i=l}^{mid}(x-x_i)$$

- 时间复杂度也是 $O(N\log^2 N)$。

洛谷 P5850 calc 加强版

- 题目链接
- 给定 m 和 k。
- 一个长度为 n 的序列 $(a_1, a_2, \ldots, a_n)$ 是合法的，当且仅当：
  1. $a_1, a_2, \ldots, a_n$ 都是 $[1, k]$ 的整数；
  2. $a_1, a_2, \ldots, a_n$ 互不相等。
- $\forall 1 \le n \le m$，求

$$\sum_{(a_1,\ldots,a_n) \text{ 合法}} \prod_{i=1}^{n} a_i \bmod 998244353$$

- $m \le 5 \times 10^5$，$1 \le m \le k \le 998244352$（需要 $O(m \log m)$ 的算法）。

洛谷 P5850 calc 加强版题解[5]

- 因为 $a_1, \ldots, a_n$ 互不相等，所以可以先计算出 $a_1, \ldots, a_n$ 单调严格递增的答案，最后乘上 $n!$。
- 现在 n 的答案为 $[x^n] \prod_{i=1}^{k} (1+ix)$。所以只要求 $\prod_{i=1}^{k} (1+ix) \bmod x^{m+1}$。

$$\begin{aligned}\prod_{i=1}^{k}(1+ix) &= \exp\left(\sum_{i=1}^{k}\ln(1+ix)\right)\\ &= \exp\left(\sum_{i=1}^{k}\sum_{j=1}^{+\infty}\frac{(-1)^{j-1}i^j x^j}{j}\right)\\ &= \exp\left(\sum_{j=1}^{+\infty}x^j\cdot\frac{(-1)^{j-1}}{j}\sum_{i=1}^{k}i^j\right)\end{aligned}$$

洛谷 P5850 calc 加强版题解[5]

- 现在瓶颈在于求 $\sum_{i=1}^{k} i^j\ (1 \le j \le m)$，这是一个典型的自然数幂求和问题。

$$\sum_{j=0}^{+\infty} x^j \cdot \frac{1}{j!} \sum_{i=1}^{k} i^j = \sum_{i=1}^{k} \sum_{j=0}^{+\infty} \frac{(ix)^j}{j!}$$

$$= \sum_{i=1}^{k} \exp(ix) = \frac{\exp((k+1)x) - \exp(x)}{\exp(x) - 1}$$

- 所以

$$\sum_{i=1}^{k} i^j = j![x^j] \frac{\exp((k+1)x) - \exp(x)}{\exp(x) - 1} = j![x^j] \frac{\frac{\exp((k+1)x) - \exp(x)}{x}}{\frac{\exp(x) - 1}{x}}$$

多项式求逆即可。

多项式 exp 的意义

问题

- N 个带标号的球分到任意多个不带标号的箱子里。每种箱子至多容 M 个求。
- 给定 $f_1, f_2, \ldots, f_M$
- 假设一种分法的箱子个数为 m，每个箱子中球的个数为 $a_1, a_2, \ldots, a_m$，那么设该分法的权值为 $\prod_{i=1}^{m} f_{a_i}$。
- 求所有分法的权值之和。

- 假设只有 $f_k \neq 0$，其它的 f 都是 0，也就是每个箱子的大小只能为 k。
- 设球的数量为 n $(k \mid n)$，那么这些方案的权值之和是 $\frac{n!}{(k!)^{n/k}(n/k)!} f_k^{n/k}$，也就是 $n![x^n] \exp\left(\frac{f_k}{k!} x^k\right)$。

多项式 exp 的意义

- 现在加上不同大小的箱子。假设大小为 k 的箱子有 a_k 个。
- 答案为

$$\sum_{(a_1,\ldots,a_M),\sum_{i=1}^{M} ia_i=N} \binom{N}{a_1, 2a_2, \ldots, Ma_M} \prod_{i=1}^{M} (ia_i)![x^{ia_i}] \exp\left(\frac{f_i}{i!}x^i\right)$$

$$= N! \sum_{\prod_{i=1}^{M} ia_i=N} [x^{ia_i}] \exp\left(\frac{f_i}{i!}x^i\right)$$

$$= N![x^N] \prod_{i=1}^{M} \exp\left(\frac{f_i}{i!}x^i\right) = N![x^N] \exp\left(\sum_{i=1}^{M} \frac{f_i}{i!}x^i\right)$$

点标号简单无向连通图计数

问题

有 N 个带标号的点，求简单无向连通图个数。

1. 简单：没有自环（自己到自己的边），没有重边（不会有两条边连接完全相同的点）。
2. 无向：边没有规定方向。
3. 连通：对于图中任意一对点，有若干条边将这两个点连接起来。

- 先忽略“连通”的要求。简单无向图，就是决定任意一对点之间有没有边，因此 n 个点的简单无向图个数为 $2^{\binom{N}{2}}$。
- 简单无向图可以拆分成若干个简单无向连通图，且这些连通图之间没有边。

点标号简单无向连通图计数

- 所以把 N 个带标号的点放到若干个箱子中，每个箱子对应一个连通图，一个装了 m 个点的箱子对应权值为 m 个点的简单无向连通图个数。所有方案的权值之和就是简单无向图个数。
- 这就与多项式 $\exp$ 的意义对应。简单无向连通图个数 EGF 的 $\exp$ 就是简单无向图个数 EGF。求个 $\ln$，最终答案为 $N![x^N]\ln\left(\sum_{n\geq 0}\frac{2^{\binom{n}{2}}}{n!}x^n\right)$

线性常系数齐次递推

- $f_n = \sum_{i=1}^{p} a_i f_{n-i}\ (n > p)$。
- 设

$$A = \begin{bmatrix} a_p & a_{p-1} & \dots & a_2 & a_1 \\ 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}$$

- 那么

$$\begin{bmatrix} a_{p+n} \\ a_{p+n-1} \\ \vdots \\ a_{n+1} \end{bmatrix} = A^n \begin{bmatrix} a_p \\ a_{p-1} \\ \vdots \\ a_1 \end{bmatrix}$$

线性常系数齐次递推

- 求 A^n 可以用矩阵快速幂做。但是这样时间复杂度是 $O(p^3 \log n)$，不能处理 p 比较大的情况。

Cayley-Hamilton 定理[6]

对于一个 n 阶方阵 A，设它的特征多项式 $p(\lambda) = \det(\lambda I_n - A)$，则 $p(A) = O$。

- 可以验证：

$$p(\lambda) = \lambda^p - \sum_{i=1}^{p} a_p \lambda^{p-i}$$

- 要求 f_n，就要求 A^{n-p}。
- 考虑 $x^{n-p} = p(x)Q(x) + R(x)$（多项式除法）。那么

$$A^{n-p} = p(A)Q(A) + R(A) = R(A)$$

线性常系数齐次递推

- $R(x)$ 是一个 $p-1$ 阶多项式。设 $R(x)=\sum_{i=0}^{p-1} r_i x^i$。

$$\begin{bmatrix} f_n \\ \vdots \\ f_{n-p+1} \end{bmatrix} = A^{n-p} \begin{bmatrix} f_p \\ \vdots \\ f_1 \end{bmatrix} = \sum_{i=0}^{p-1} r_i A^i \begin{bmatrix} f_p \\ \vdots \\ f_1 \end{bmatrix} = \sum_{i=0}^{p-1} r_i \begin{bmatrix} f_{i+p} \\ \vdots \\ f_{i+1} \end{bmatrix}$$

- 要求 f_n，只用考虑矩阵的第一行：

$$f_n = \sum_{i=0}^{p-1} r_i f_{i+p}$$

- 快速求出 $x^{n-p} \bmod p(x)$：快速幂。
  - 计算 $x^k \bmod p(x)$：若 k 为偶数，则 $x^k \equiv \left(x^{k/2}\right)^2 \pmod{p(x)}$，否则 $x^k \equiv x^{k-1}x \pmod{p(x)}$。
  - 用多项式除法，时间复杂度为 $O(p \log p \log n)$。

线性常系数齐次递推

- 快速求出 $f_{i+1}, \ldots, f_{i+p-1}$：
  - f 的 OGF：

$$F(x) = \sum_{k=1}^{+\infty} f_k x^k$$

$$\forall n \le p,\ f_n - \sum_{i=1}^{p} a_i f_{n-i} = 0$$

$$F(x)\left(1 - \sum_{i=1}^{p} a_i x^i\right) = \sum_{i=1}^{p} x^i \left(f_i - \sum_{j=1}^{i-1} a_j f_{i-j}\right)$$

  - 两边同除 $1 - \sum_{i=1}^{p} a_i x^i$，多项式求逆，时间复杂度 $O(p \log p)$。
- 总时间复杂度为 $O(p \log p \log n)$。

求一类微分方程的特解

问题

求

$$\sum_{i=0}^{n} a_i y^{(i)} = e^{\alpha x} \left(\sum_{j=0}^{m} b_j x^j \right)$$

形如 $e^{\alpha x}$ 乘上一个多项式的特殊解（$a_n \neq 0$）。

- 先考虑 $y = e^{\alpha x} x^k$ 的导数。
- 记

$$k^{\underline{r}} = \begin{cases} \prod_{i=0}^{r-1} (k-i) & (r > 0) \\ 1 & (r = 0) \end{cases}$$

称为 k 的 r 次下降幂。

求一类微分方程的特解

- 观察

$$
\begin{aligned}
y' &= e^{\alpha x}\left(\alpha x^k + k x^{k-1}\right), \\
y^{(2)} &= e^{\alpha x}\left[\alpha^2 x^k + 2\alpha k x^{k-1} + k^{\underline{2}} x^{k-2}\right], \\
y^{(3)} &= e^{\alpha x}\left[\alpha^3 x^k + 3\alpha^2 k x^{k-1} + 3\alpha k^{\underline{2}} x^{k-2} + k^{\underline{3}} x^{k-3}\right]
\end{aligned}
$$

- 可以发现规律：

$$
y^{(r)} = e^{\alpha x} \sum_{i=0}^{r} \binom{r}{i} \alpha^{r-i} k^{\underline{i}} x^{k-i}
$$

- 可以数学归纳法证明。

求一类微分方程的特解

- 归纳法证明。
- $r = 0$ 时显然成立。
- 假设 $y^{(r)} = e^{\alpha x} \sum_{i=0}^{r} \binom{r}{i} \alpha^{r-i} k^i x^{k-i}$。

$$
\begin{aligned}
y^{(r+1)} &= e^{\alpha x} \cdot \alpha \sum_{i=0}^{r} \binom{r}{i} \alpha^{r-i} k^i x^{k-i} + e^{\alpha x} \cdot \sum_{i=0}^{r} \binom{r}{i} \alpha^{r-i} k^i \cdot (k-i) x^{k-i-1} \\
&= e^{\alpha x} \left[\sum_{i=0}^{r} \binom{r}{i} \alpha^{r+1-i} k^i x^{k-i} + \sum_{i=1}^{r+1} \binom{r}{i-1} \alpha^{r+1-i} k^{i-1} (k-(i-1)) x^{k-i} \right] \\
&= e^{\alpha x} \left[\sum_{i=0}^{r+1} \binom{r}{i} \alpha^{r+1-i} k^i x^{k-i} + \sum_{i=1}^{r+1} \binom{r}{i-1} \alpha^{r+1-i} k^i x^{k-i} \right] \\
&= e^{\alpha x} \sum_{i=0}^{r+1} \binom{r+1}{i} \alpha^{r+1-i} k^i x^{k-i}.
\end{aligned}
$$

求一类微分方程的特解

- 实际上这个系数 $\binom{r}{i}\alpha^{r-i}k^{\underline{i}}$ 也可以用组合意义推出来。
- $\left(e^{\alpha x}x^k\right)' = e^{\alpha x}\left(kx^{k-1} + \alpha x^k\right)$。所以对于 $e^{\alpha x}x^k$，求导后有两种可能：
  1. x 的次数减 1，也就是 $x^k \to x^{k-1}$，系数乘上 k；
  2. x 的次数不变，$x^k \to x^k$，系数乘上 α。
- 经过 r 次求导操作后，$x^k \to x^{k-i}$，说明有 i 次操作 1，$r-i$ 次操作 2。
- 对于操作 1，这些系数乘积为 $k(k-1)\dots(k-i+1) = k^{\underline{i}}$。
- 对于操作 2，这些系数成绩为 α^{r-i}。
- 从 r 中选 i 个操作 1，有 $\binom{r}{i}$ 种选法。
- 所以系数为 $\binom{r}{i}\alpha^{r-i}k^{\underline{i}}$。

求一类微分方程的特解

- 所以

$$\sum_{i=0}^{n} a_i y^{(i)} = e^{\alpha x} \sum_{j=0}^{n} k^j x^{k-j} \sum_{r=0}^{n} \binom{r}{j} \alpha^{r-j} a_r$$

- 设

$$C_j = \sum_{i=j}^{n} \binom{i}{j} \alpha^{i-j} a_i$$

- 所以

$$\sum_{i=0}^{n} a_i y^{(i)} = e^{\alpha x} \sum_{j=0}^{k} k^j C_j x^{k-j}$$

- 枚举 $l = k - j$:

$$\sum_{i=0}^{n} a_i y^{(i)} = e^{\alpha x} \sum_{l=0}^{k} k^{k-l} C_{k-l} x^l$$

求一类微分方程的特解

- 我们取最小的 p 使得 $C_p \neq 0$。
- 那么当 $y = e^{\alpha x} x^k$ 时，$\sum_{i=0}^{n} a_i y^{(i)}$ 中 x^k 项到 x^{k-p+1} 项系数都为 0，x^{k-p} 项系数不为 0。
- 所以要求该微分方程的解，最高次项应该为 $e^{\alpha x} x^{m+p}$。
- 我们从高位到低位逐位确定，

$$\sum_{i=0}^{n} a_i (e^{\alpha x} x^{m+p})^{(i)} = e^{\alpha x} \sum_{l=0}^{m} (m+p)^{\underline{m+p-l}} C_{m+p-l} x^l$$

- 为了让 $\sum_{i=0}^{n} a_i y^{(i)}$ 的 $e^{\alpha x} x^m$ 系数为 b_m，$e^{\alpha x} x^{m+p}$ 系数应为 $\frac{b_m}{(m+p)^{\underline{p}} C_p}$。

求一类微分方程的特解

$$\frac{b_m}{(m+p)^{\underline{p}}C_p}e^{\alpha x}x^{m+p}$$

- 接下来的 b_i $(i<m)$ 应该减掉这一项对它的影响，即

$$b_i \leftarrow b_i - \frac{b_m}{(m+p)^{\underline{p}}C_p}C_{m+p-i}(m+p)^{\underline{m+p-i}}$$

- 对于新的微分方程，继续求解。
- 设 $z_p,\ldots,z_{m+p}$ 分别为 $e^{\alpha x}x^p,\ldots,e^{\alpha x}x^{m+p}$ 项系数。
- 则

$$z_i = \frac{b_{i-p} - \sum_{j=i+1}^{m+p} z_j C_{j-i+p} j^{\underline{j-i+p}}}{C_p i^{\underline{p}}}$$

$$= \frac{1}{C_p i^{\underline{p}}}\left(b_{i-p} - \frac{1}{(i-p)!}\sum_{j=i+1}^{m+p}(z_j\cdot j!)\cdot C_{j-i+p}\right)$$

- 可以 CDQ 分治 FFT。

求一类微分方程的特解

- 未使用 FFT 的 Python 代码
- 也可以用类似的方法求 $\sum_{i=0}^{n} a_i y^{(i)} = P(x)\sin(\alpha x) + Q(x)\cos(\alpha x)$ 的特解，但是要注意 $\sin(\alpha x)$ 求导后会影响 $\cos$，反之亦然。

参考文献 I

[1] Shun HUNG H.Architecture of 8-point Decimation-in-time FFT. [Online; created 16:58, 27 June 2011]. 2011. https://commons.wikimedia.org/wiki/File:DIT_N8_ALL.JPG.

[2] Wikipedia.Master theorem (analysis of algorithms) — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Master%20theorem%20(analysis%20of%20algorithms)&oldid=1122572444. [Online; accessed 30-November-2022]. 2022.

[3] 毛啸. 再探快速傅里叶变换. 2016 年信息学奥林匹克竞赛中国国家队候选队员论文, 2016.

[4] 陈丹琦. 从《Cash》谈一类分治算法的应用. 2008 年信息学国家集训队作业, 2008.

参考文献 II

[5] lostream.iostream 的博客 - 题解 P5850【calc 加强版】. 2020. https://www.luogu.com.cn/blog/user13052/solution-p5850.

[6] Wikipedia.Cayley–Hamilton theorem — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Cayley%E2%80%93Hamilton%20theorem&oldid=1120554488. [Online; accessed 01-December-2022]. 2022.

谢 谢